CAB403 Assignment Report

Process Management and Distributed Computing

Student Name: Manan Patel
Student Number: n9839950

Student Name: Min-Pu Tsai
Student Number: n930049

A: Statement of Completeness

Assignment Specification: We have attempted task 1, take 2 and task 3. Other than that game works the same as specification.

Problem: The toughest part of the assignment was thread pool (task 3). Allowing the system to use 10 users as the same time, with help of . Attempt has been made to do thread pool and . The other challenging part was connect the server and client with authentication.txt provided.

Deficiency: The leaderboard shows that a particular player as having playing X games and won Y of them. But you will notice that the leaderboard also shows the duplicate record for the same player. Cleary we could managed to do to both simultaneously. If the leaderboard.c include the commented-out section where the player has X games won and Y games lost is implement. It has been commented out because of the contradiction.

B: Information About Team

Manan Patel – n9839950
Min-Pu Tsai – n9300449

C: Statement of Contributions

Both the students have similar contribution.
Manan takes the server-side in which port connection, detection and verification of the input from the user (Authentication.txt).
Min-Pu complete the client –side, multithreaded implementation with game play and adding request with getting request.

D: Data Structures of Playfield

```
void play(long sock) {
        struct Board *board = create_board();
        char *coords; int choice = 0;
        int16_t response;
}
```

E: Data Structure of Leaderboard

```
/**
  * A leaderboard records.
  *
  * Maintaining a sorted linked list is a comparatively difficult last.
  * similarly, a linked list is very hard to sort. Arrays on the other hand
  * can be sorted easily with the system sort (qsort).
  *
  * Thus, our leaderboard is an array of Record objets as for the type
  * defined below. We display up to LEADERBOARD_SIZE records. However, we have
  * space for LEADERBOARD_RECORD + 1 records in storage. After an element
  * is added, the list is sorted and the last item on the list is deleted.
  */
struct Record {
   char name[LEADERBOARD_NAME_LENGTH];
   int seconds;
   int played;
   int won;
};
```

F: Description of Handling Critical-section Problem

The part which was most critical was task 3 thread pool. It needed a great skill to complete that part of the assignment. Till the last day of the assignment we were struggling with thread pool and handling all the request around the file
Handline the request in loop of the threadpool.c , which is infinite loop of request handling seems to be toughest of all .

G: Description of Create & Management in Thread Pool

```
/* global mutex for our program. assignment initializes it. */
/* note that we use a RECURSIVE mutex, since a handler      */
/* thread might try to lock it twice consecutively.         */
/*
 * function add_request(): add a request to the requests list
 * algorithm: creates a request structure, adds to the list, and
 *            increases number of pending requests by one.
 * input:     request number, linked list mutex.
 * output:    none.
 */
/*
 * function get_request(): gets the first pending request from the requests
list
 *                         removing it from the list.
 * algorithm: creates a request structure, adds to the list, and
 *            increases number of pending requests by one.
 * input:     request number, linked list mutex.
 * output:    pointer to the removed request, or NULL if none.
 * memory:    the returned request need to be freed by the caller.
 */
/*
 * function handle_requests_loop(): infinite loop of requests handling
 * algorithm: forever, if there are requests to handle, take the first
 *            and handle it. Then wait on the given condition variable,
 *            and when it is signaled, re-do the loop.
 *            increases number of pending requests by one.
 * input:     id of thread, for printing purposes.
 * output:    none.
 */
```

H: Instructions of Compiling & Running

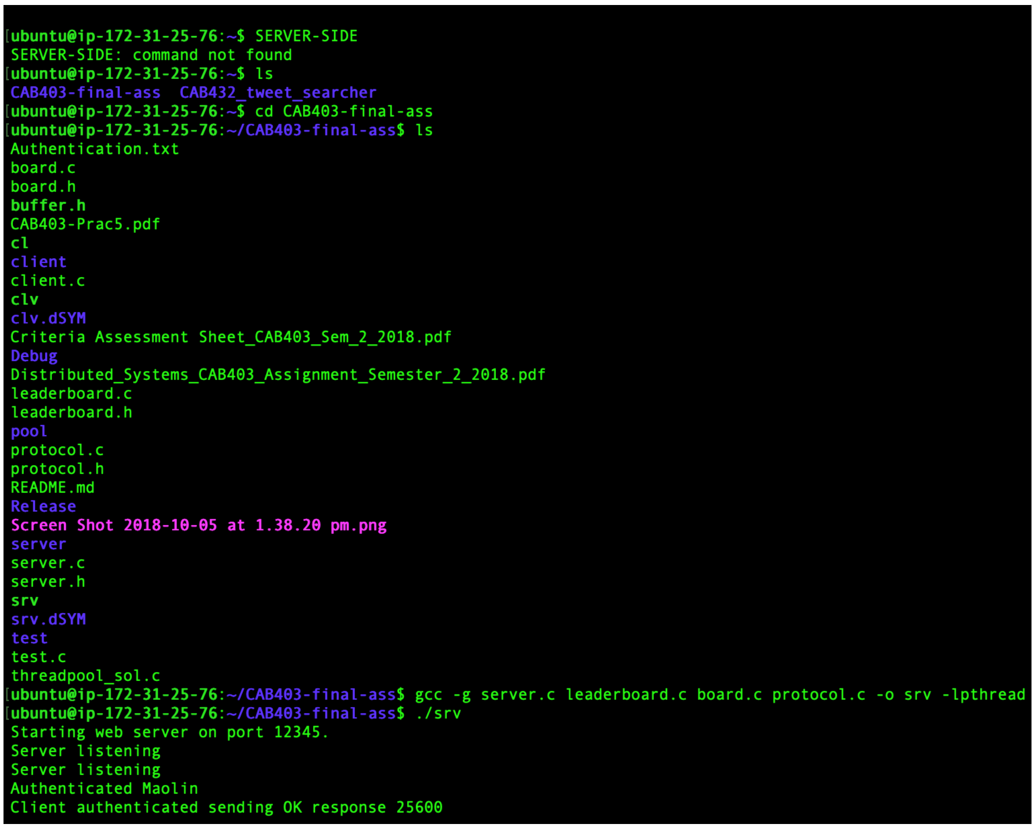
First of all, we need two bash/terminal. One as service to receive order and respond, and another as client who request and process game play. To apply which is server and another is client, we use: `$ SERVER-SIDE` and `$ CLIENT-SIDE`
When it comes to the compiling. For service side, it is:

```
$ gcc -g server.c leaderboard.c board.c protocol.c -o srv -lpthread
```

And it's client side:

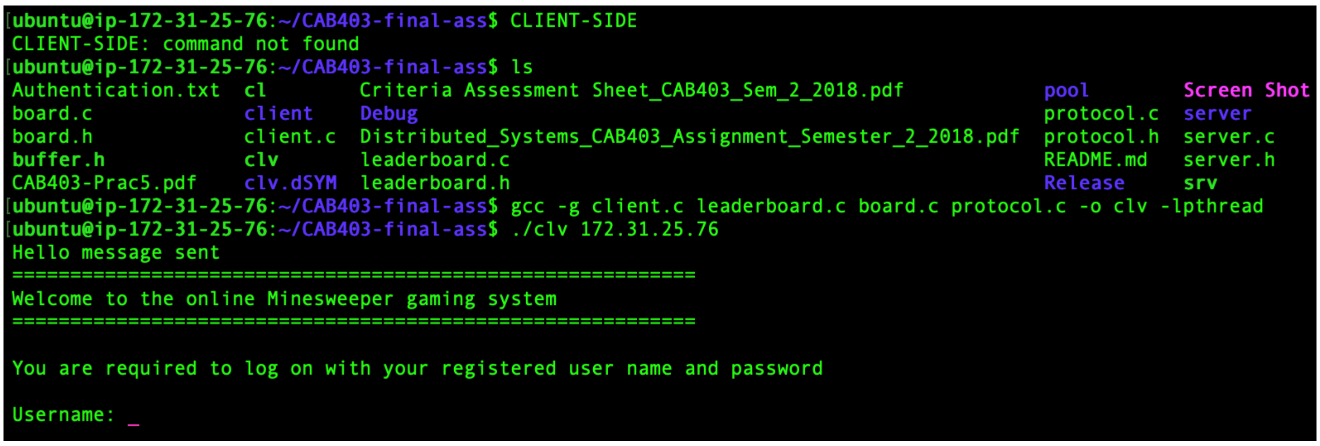
```
$ gcc -g client.c leaderboard.c board.c protocol.c -o clv -lpthread
```

1. Server side, compiling and running, first of two bash/terminal (notice that server have to run first): `$ ./srv`



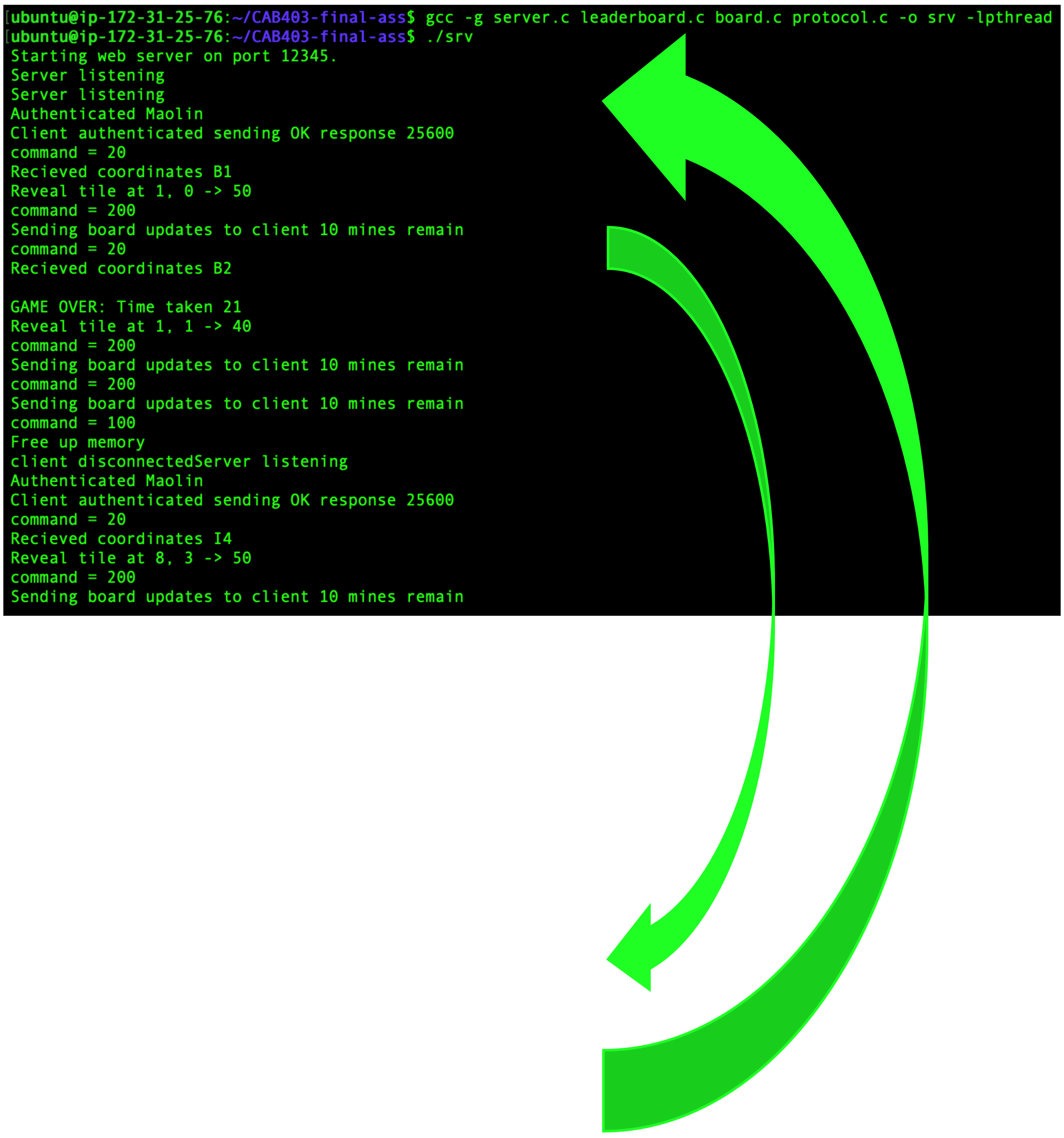
```
ubuntu@ip-172-31-25-76:~$ SERVER-SIDE
SERVER-SIDE: command not found
ubuntu@ip-172-31-25-76:~$ ls
CAB403-final-ass  CAB432_tweet_searcher
ubuntu@ip-172-31-25-76:~$ cd CAB403-final-ass
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ ls
Authentication.txt
board.c
board.h
buffer.h
CAB403-Prac5.pdf
cl
client
client.c
clv
clv.dSYM
Criteria Assessment Sheet_CAB403_Sem_2_2018.pdf
Debug
Distributed_Systems_CAB403_Assignment_Semester_2_2018.pdf
leaderboard.c
leaderboard.h
pool
protocol.c
protocol.h
README.md
Release
Screen Shot 2018-10-05 at 1.38.20 pm.png
server
server.c
server.h
srv
srv.dSYM
test
test.c
threadpool_sol.c
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ gcc -g server.c leaderboard.c board.c protocol.c -o srv -lpthread
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ ./srv
Starting web server on port 12345.
Server listening
Server listening
Authenticated Maolin
Client authenticated sending OK response 25600
```

2. Client side, compiling and running, second of two bash/terminal: `$ ./clv`

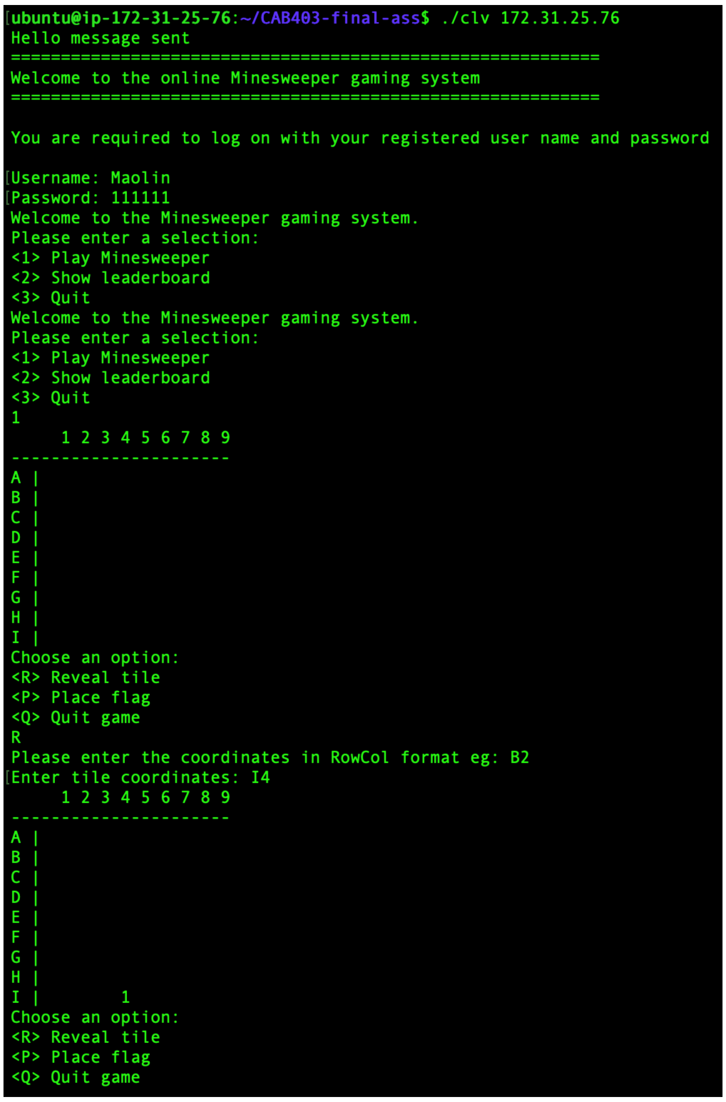


```
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ CLIENT-SIDE
CLIENT-SIDE: command not found
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ ls
Authentication.txt  cl        Criteria Assessment Sheet_CAB403_Sem_2_2018.pdf            pool        Screen Shot
board.c             client    Debug                                                      protocol.c  server
board.h             client.c  Distributed_Systems_CAB403_Assignment_Semester_2_2018.pdf  protocol.h  server.c
buffer.h            clv       leaderboard.c                                              README.md   server.h
CAB403-Prac5.pdf    clv.dSYM  leaderboard.h                                              Release     srv
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ gcc -g client.c leaderboard.c board.c protocol.c -o clv -lpthread
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ ./clv 172.31.25.76
Hello message sent
============================================================
Welcome to the online Minesweeper gaming system
============================================================

You are required to log on with your registered user name and password

Username: _
```

3. Conversation between server and client:

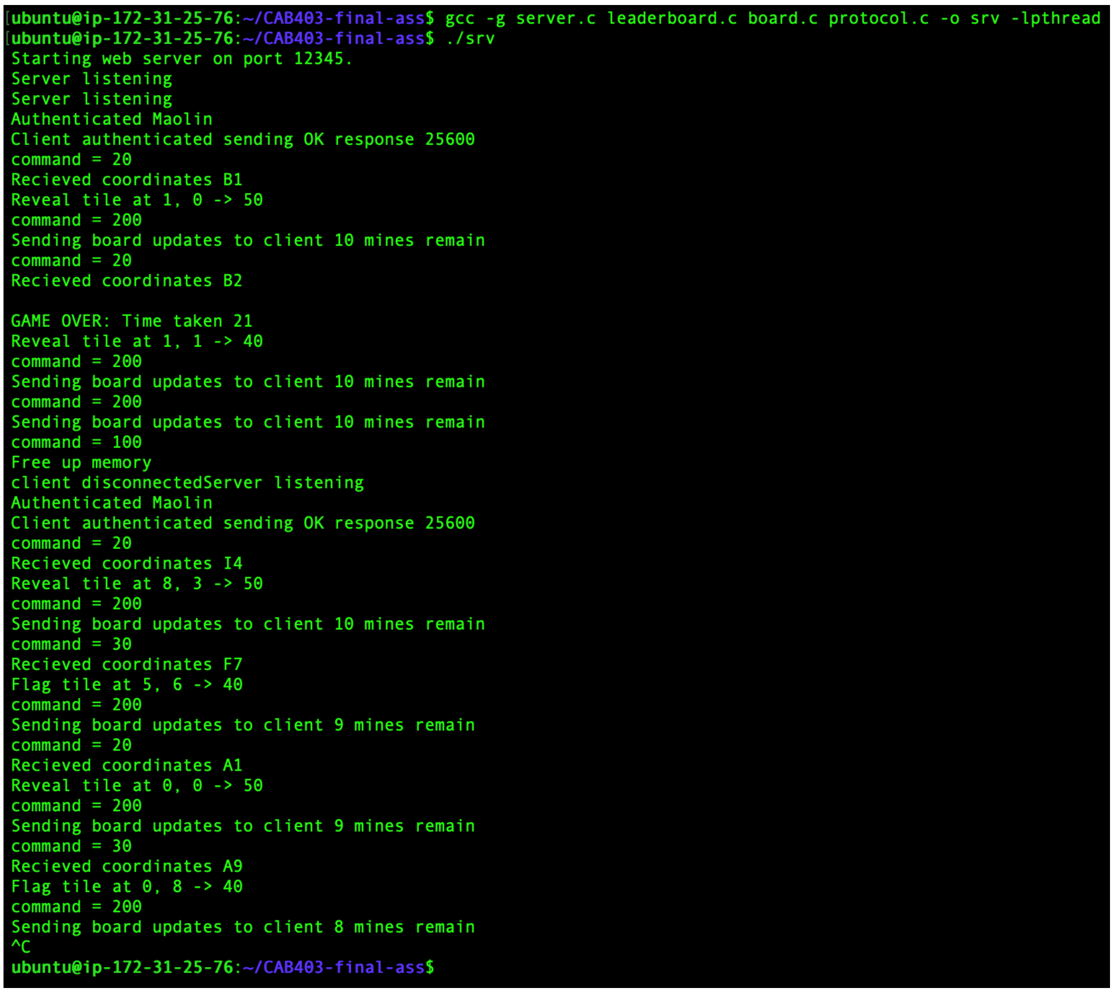


```
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ gcc -g server.c leaderboard.c board.c protocol.c -o srv -lpthread
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ ./srv
Starting web server on port 12345.
Server listening
Server listening
Authenticated Maolin
Client authenticated sending OK response 25600
command = 20
Recieved coordinates B1
Reveal tile at 1, 0 -> 50
command = 200
Sending board updates to client 10 mines remain
command = 20
Recieved coordinates B2

GAME OVER: Time taken 21
Reveal tile at 1, 1 -> 40
command = 200
Sending board updates to client 10 mines remain
command = 200
Sending board updates to client 10 mines remain
command = 100
Free up memory
client disconnectedServer listening
Authenticated Maolin
Client authenticated sending OK response 25600
command = 20
Recieved coordinates I4
Reveal tile at 8, 3 -> 50
command = 200
Sending board updates to client 10 mines remain
```



```
[ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ ./clv 172.31.25.76
Hello message sent
===========================================================
Welcome to the online Minesweeper gaming system
===========================================================

You are required to log on with your registered user name and password

[Username: Maolin
[Password: 111111
Welcome to the Minesweeper gaming system.
Please enter a selection:
<1> Play Minesweeper
<2> Show leaderboard
<3> Quit
Welcome to the Minesweeper gaming system.
Please enter a selection:
<1> Play Minesweeper
<2> Show leaderboard
<3> Quit
1
     1 2 3 4 5 6 7 8 9
----------------------
A |
B |
C |
D |
E |
F |
G |
H |
I |
Choose an option:
<R> Reveal tile
<P> Place flag
<Q> Quit game
R
Please enter the coordinates in RowCol format eg: B2
[Enter tile coordinates: I4
     1 2 3 4 5 6 7 8 9
----------------------
A |
B |
C |
D |
E |
F |
G |
H |
I |        1
Choose an option:
<R> Reveal tile
<P> Place flag
<Q> Quit game
```

4. Press ctrl + C to exit the server:



```
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ gcc -g server.c leaderboard.c board.c protocol.c -o srv -lpthread
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$ ./srv
Starting web server on port 12345.
Server listening
Server listening
Authenticated Maolin
Client authenticated sending OK response 25600
command = 20
Recieved coordinates B1
Reveal tile at 1, 0 -> 50
command = 200
Sending board updates to client 10 mines remain
command = 20
Recieved coordinates B2

GAME OVER: Time taken 21
Reveal tile at 1, 1 -> 40
command = 200
Sending board updates to client 10 mines remain
command = 200
Sending board updates to client 10 mines remain
command = 100
Free up memory
client disconnectedServer listening
Authenticated Maolin
Client authenticated sending OK response 25600
command = 20
Recieved coordinates I4
Reveal tile at 8, 3 -> 50
command = 200
Sending board updates to client 10 mines remain
command = 30
Recieved coordinates F7
Flag tile at 5, 6 -> 40
command = 200
Sending board updates to client 9 mines remain
command = 20
Recieved coordinates A1
Reveal tile at 0, 0 -> 50
command = 200
Sending board updates to client 9 mines remain
command = 30
Recieved coordinates A9
Flag tile at 0, 8 -> 40
command = 200
Sending board updates to client 8 mines remain
^C
ubuntu@ip-172-31-25-76:~/CAB403-final-ass$
```